8-2010

Subdivision-Based 3D Remeshing With a Fast Spherical Parameterization Method

Ruqin Zhang
Iowa State University

Eliot H. Winer
Iowa State University, ewiner@iastate.edu

James H. Oliver
Iowa State University, oliver@iastate.edu

Proceedings of the ASME 2010 International Design Engineering Technical Conferences &
Computers and Information in Engineering Conference
IDETC/CIE 2010
August 15-18, 2010, Montreal, Quebec, Canada

DETC2010-28904

SUBDIVISION-BASED 3D REMESHING WITH A FAST SPHERICAL PARAMETERIZATION METHOD

Ruqin Zhang
Research Assistant
Virtual Reality Applications Center
Dept. of Mechanical Engineering
Iowa State University
Ames, Iowa 50010-2274
ruzhang@iastate.edu

Eliot Winer
Associate Professor
Virtual Reality Applications Center
Dept. of Mechanical Engineering
Iowa State University
Ames, Iowa 50010-2274
ewiner@iastate.edu

James H. Oliver
Professor
Virtual Reality Applications Center
Dept. of Mechanical Engineering
Iowa State University
Ames, Iowa 50010-2274
oliver@iastate.edu

ABSTRACT

3D mesh parameterization is widely investigated with various parameter domains and applied in many computer graphics applications. As many surface meshes are manifolds of genus zero, mapping these meshes onto a topologically equivalent sphere provides some advantages. We introduce an efficient parameterization method based on barycentric embedding for this spherical domain. This method provides an overlapping solution which emphasizes on eliminating the vertex overlappings to ensure bijectivity. Experimental results indicate that it works faster than existing spherical parameterization methods. And we also provide a robust spherical remeshing algorithm based on spherical mesh subdivision. A local recursive subdivision process is employed to cover all the geometric details from the original mesh. Such subdivision process can be controlled to match the desired level of details (LOD), which will create a group of mesh representations with different resolutions. This multi-resolution remeshing framework could benefit various graphical applications including geometry rendering, mesh simplification/refinement, model morphing and etc.

Keywords: mesh parameterization, remeshing, subdivision, multi-resolution

INTRODUCTION

3D mesh parameterization is a powerful technique for geometric modeling and processing in numerous applications of computer graphics. Considering any two surfaces with similar topology, usually there exists a one-to-one (bijective) mapping between them. In general, if one of such surfaces is represented by a triangle-based mesh, the mapping process is referred to as mesh parameterization (1, 2, 3). Typically, the destination surface that the mesh is mapped to is called the parameter domain. And the other one is treated as the source mesh. The objective of mesh parameterization is to generate a map between such source mesh and a triangulation of the domain. An essential goal of parameterization is to get a bijective map, where for each vertex in the source mesh there is one and only one correspondent vertex in the target parameterization domain.

Mesh parameterization was originally investigated and introduced into the fields of computer graphics and computational geometry as a technique for texture mapping. With fast growth in geometry related field, more and more research integrated and benefited from mesh parameterization. It has been developed and widely applied to various applications. Such fields that benefit from parameterization include texture and detail mapping, detail creation and synthesis, geometry interpolation and morphing, mesh fixing and manipulation, mesh compression, remeshing, medical imaging and visualization, and etc.

The traditional approaches for surface parameterization are mostly focused in mapping meshes with disk-like topology to a planar target domain (2). The mapping from the source mesh to parameter domain is represented by the parametric locations of vertices within the plane. Various optimization methods are applied to freely relocate the vertices within the domain as long as the mesh is maintained bijectively. While, with the development of related applications of computer graphics, the 2D planar parameterization domain does not meet the requirements well sometimes. High dimensional parameter domains become necessary and so lots of different algorithms and methodologies are developed. Based on the type of target mapping domain, most mesh parameterizations can be classified as planar parameterization, simplicial parameterization, spherical parameterization and etc.

Planar Parameterization

Early planar parameterization aimed to address the issue of texture mapping for surface with disk-like topology. While recent parameterization applications involve other surface properties (e.g. surface normal) or process other geometry operations (e.g. remeshing, morphing and etc.). Usually, parameterization from a 3D surface to a 2D planar domain unavoidably produces distortions except some rare cases. A lot of parameterization methods are developed to deal with distortion minimization in either one or more formats of: angular, stretch and area.

Angle preserving parameterization would be one of the most investigated methods in this field. Eck et al. (4) develop the harmonic or cotangent weights parameterization. To ensure bijectivity, Kharevych et al. (5) introduce intrinsic Delaunay triangulation of the surface mesh as an input to the harmonic parameterization. The anisotropic mesh parameterization scheme (6) brings in an anisotropic modification to Floater's shape preserving parameterization method (7). Lee et al. (8) create a fixed virtual boundary to make the real domain boundary free and thus the real boundary can better reflect the shape of the source 3D boundary. With such freely moving boundary, the parameterization is able to introduce less distortion than methods with fixed boundary. Similarly, Zhang et al. (9) present an automatic planar parameterization method for mesh segmentation and flattening. LSCM (Least Squares Conformal Maps) (10) and DCP (Discrete Conformal Parameterization) (11) are two explicit formulations for linear parameterization with free boundary. They aim to minimize angular distortions, while are independently proposed with different formulations of harmonic energy. The ABF and ABF++ (Angle-Based Flattening) (12, 13) introduce a novel method with a definition in term of angles. The algorithm runs iterations to search for angles that are as close as possible to the angles in the original 3D source mesh. Zayer et al. (14, 15) extend ABF with additional methods borrowed from traditional parameterization process in terms of vertex coordinates. They employ an iterative free-boundary conformal method to minimize distortions.

Distance preserving parameterization is another optimization method. Lévy and Mallet (16) introduce a technique for non-distorted texture mapping which allows local distortion minimization in order of user's preference. Sander et al. (17) present a technique of constructing a progressive mesh and introduce two metrics of parameterization stretch, which are widely used for linear distortion comparison between different mapping methods. Iso-charts (18) merge stretch-minimizing parameterization and the multi-dimensional scaling (MDS) parameterization to create texture atlas for arbitrary meshes. Sander et al. (19) extend the method with signal specialized parameterization, which allows the user to affect the distribution of distortions along the mesh surface. And Tewari et al. (20) report a more accurate signal with significant savings in texture area than the signal specialized parameterization.

Area preserving parameterization, referred as authalic, work with area preservation for mesh triangles by typically introducing additional optimization terms or constraints. Desbrun et al. (11) derive a similar method from their discrete conformal map (DCP) algorithm and implement a linear formulation for local triangular area preservation. Their formulation supports a tradeoff between angular and area distortions. While Degener et al. (21) directly target at global area deformation for mesh parameterization. A non-linear formulation is developed with an energy functional which measures angular and area distortions simultaneously with a tradeoff parameter controlled by users.

Planar parameterization techniques are only applicable to 3D meshes with same topology of disk. For closed meshes or high genus meshes, some techniques have been developed to cut the source mesh into an atlas of charts before parameterization. Maillot et al. (22) introduce an algorithm to automatically produce an atlas from any type of mesh for texture mapping. Multi-chart geometry images (23) utilize a representation for arbitrary geometric surfaces to map the surface piecewise onto charts of arbitrary shape by using an atlas construction. Gu and Yau (24) present a global parameterization algorithm which preserves conformality and introduce no boundary discontinuities by constructing a basis of the underlying linear solution space. PolyCube-Maps (25) have been introduced as a technique to map 3D meshes on a set of square charts. Instead of segmenting source mesh into separate patches, another widely used cutting technique is to cut it into a single chart. Sheffer et al. (26) introduce a fast technique to cut the surface areas with high surface curvature, which is proven to produce less distortion and visual distraction. Sorkine et al. (27) provide the first method to parameterize and partition the mesh simultaneously and automatically. Lazarus et al. (28) present two optimal algorithms for the problem related to cutting the surface with high genus and map it into a topological disk from canonical polygonal schema. A handle cutting method by Erickson et al. (29) aims to converting a polyhedral manifold surface into a single topological disk by minimizing either the total number of cutting edges or total cutting length. And Ni et al. (30) perform small number of cuttings for genus reduction by solving a relaxed form of Laplace's equation to find a fair Morse function (31) with a user-controlled number and configuration of critical points.

Simplicial Parameterization

Mesh segmentation and seam cutting will inevitably generate discontinuities during parameterization, which is intolerable for some sensitive graphic applications. Simplicial complex has been widely utilized as parameterization domain to avoid unnecessary segmentation or cutting.

The process of simplicial parameterization consists of defining a coarse simplicial complex and mapping the source mesh to the base complex domain by computing its barycentric coordinates. Eck et al. (4) introduce a method to convert completely arbitrary meshes to multiresolution form by

overcoming the subdivision connectivity restriction with the construction of parameterization over a simplicial complex domain. Hybrid mesh (32) is another multiresolution surface representation with both regular and irregular refinements. Praun et al. introduce a parameterization algorithm (33) which establishes consistent parameterizations for a group of models by sharing the same base domain and respect features. They implement remeshing based on the same connectivity, which forms a wide range of application algorithms including principal component analysis (PCA), wavelet transforms, detail and texture transfer between models and etc. Khodakovsky et al. (34) set up a global system where the adjacent domain faces are regarded as they are locally opened up into a plane, which can be solved with convergence in a fast manner. Schreiner et al. (35) construct the simplicial complex domain by making use of a set of correspondences between feature vertices from the input meshes as the vertices to form the base domain. In a similar approach, cross-parameterization (36) also preserves the feature vertex correspondences from user input and the shape correlation between the models. Boier-Martin et al. (37) develop a method to parameterize irregular triangular meshes over polyhedral domains with quadrilateral faces with construction of a coarse mesh from region boundaries.

Spherical Parameterization

Spherical parameterization domain is similar to (may be viewed as a special case of) simplicial complex domain. Topologically, close manifold genus-0 meshes are equivalent to a sphere. So a spherical base surface is the natural parameterization domain for those meshes. Compared to the planar and simplicial domains, the advantage of the spherical parameterization is that it allows smooth, seamless and continuous parameterization of genus zero models.

Reduce the 3D spherical parameterization down to the 2D planar case is on the top of the head for some researchers (38). They cut the closed mesh into two pieces and parameterize them onto a 2D planar domain with the same fixed boundary. Then map each disk to a hemisphere and combine these two hemispheres into a full sphere. Haker et al. (39) pick one triangle as a boundary and compute a planar parameterization of the rest open mesh over the triangle by applying planar parameterization method, and finally calculate the stereographic projection to obtain the spherical parameterization. Zayer et al. (40) cut the mesh along a date line defined by some poles from user input and apply planar parameterization over a rectangular domain by solving a Laplace equation in curvilinear coordinates.

Directly mapping a mesh over spherical domain is more natural than applying 2D disk as temporary median domain. Similar to the work by Das and Goodrich (41), Shapiro and Tal (42) apply mesh simplification for vertices removal until a tetrahedron remains. They then map the tetrahedron onto the closed domain followed by adding removed vertices back one by one. Birkholz (43) provides shape-preserving parameterization method with mesh simplification. Edge collapse method with a collapse order of edge length is utilized to obtain the tetrahedron.

Barycentric based convex boundary methods have been well developed in planar parameterization and extended to spherical base domain with Gauss-Seidel iterations. Kobbelt et al. (44) borrow the shrink wrapping process by adapting the deformable surface technique from image processing and apply it in parameterization. Alexa (45, 46) performs heuristic iterative procedures with uniform weights of spherical parameterization for genus 0 polyhedral. From spectral graph theory and its extension, Gotsman et al. (47) generalize the method of barycentric coordinates for planar parameterization to solve the spherical mapping problem. However, they don't provide an effect solution to this quadratic system. Saba et al. (48) report the failure of solving these equations with simple iterative methods and introduce a successful numerical approach by using a bunch of optimization methods associated with an algebraic multi-grid technique.

Some other approaches are also developed for spherical parameterization. Praun and Hoppe (49) develop a scheme for sampling the spherical domain using uniformly subdivided polyhedral domains with the minimization of a stretch-based distortion. Sheffer et al. (50) extend the idea of ABF (Angle-based Flattening) from 2D planar parameterization to the spherical case by formulating and solving an optimization procedure in terms of angles on the sphere.

APPROACH OVERVIEW

In this work, our parameterization is specialized to closed genus-0 meshes. The method is based upon barycentric coordinates with convex boundary. Unlike most existing similar approaches which deal with each single vertex in the mesh equally, our method spends more power in solving the spikes (overlapping areas) during parameterization, which help speed up the process. The algorithm starts with normalizing the source mesh onto a unit sphere and followed by some initial relaxation from Gauss-Seidel iterations. Alexa (45, 46) keep these relaxation steps going until all the foldovers disappear. However, it is reported that such process is not guaranteed to converge. Here we introduce a different approach. After the initial relaxation step, most overlapping vertices are solved. We then provide our overlapping solution for remaining foldovers which are very hard to solve. Finally, a minimization process is also implemented to reduce the distortion. Due to its emphasis on overlapping solving, this parameterization process is faster than existing spherical mapping methods (refer to the experimental results section).

Besides spherical parameterization, we also generate a remeshing method based on spherical domain. This method is developed from the concept of spherical mesh subdivision. The local recursive subdivision can be set to match the desired LOD (level of details) for source spherical mesh. Such LOD could be controllable and it allows various outputs with different resolutions. This multi-resolution subdivision scheme then employs a triangular validation process which assures a valid

triangulation for the remeshing. And the final mesh merging and reconstruction process produces the remeshing model with desired LOD specified from user.

SPHERICAL PARAMETERIZATION

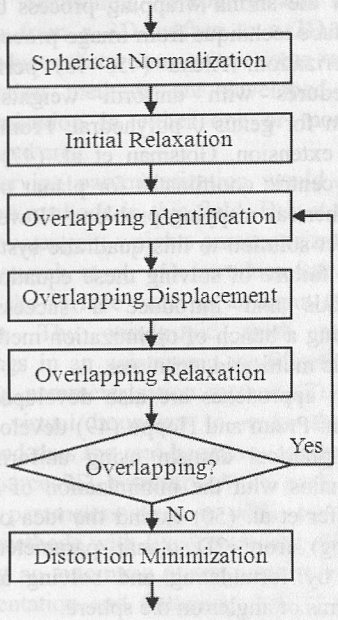


Figure 1: Flowchart of our spherical parameterization process

As reported in (48), there are two main challenges for most of existing spherical parameterization methods. The first one is the difficulty to make the procedures stable and able to converge. Sometimes, these processes will cause endless iterations or residual overlapping areas. No triangle overlapping is the most crucial requirement for a valid parameterization since a bijective one-to-one mapping is a must for many applications based on parameterization. The second challenge is the time-wise efficiency. Due to the computational complexity for some algorithms, the processes involve intense calculations so that the time spent could be up to hours or even more. These inefficient procedures will hurdle lots of applications that care about speed and pace. Our algorithm is experimented to be faster than those existing spherical parameterization methods. The flowchart of our spherical parameterization framework is shown in Figure 1.

Our algorithm targets right on the challenge of solving overlapping for most spherical parameterization. Like some of the existing methods, we employ barycentric maps with uniform weight for our embeddings and can be easily extended to other weight formats if necessary. The method consists of several procedures as shown in the diagram. These procedures include: i) Normalize and project each vertex from the source mesh onto a unit sphere; ii) Apply Gauss-Seidel iterations for the initial relaxation to solve most foldovers; iii) Identify and find remaining overlapping vertices; iv) Stretch each overlapping area and form a convex boundary for it; v) Fix these convex boundaries and map the overlapping vertices over related areas until no more overlapping exists; vi) Relax the whole spherical mesh with displacement constraint to minimize distortions.

Initial Relaxation

After a triangular mesh is loaded, it will be normalized and projected onto a unit sphere directly. At this point, such spherical mesh contains massive irregular overlappings. To make it easier for further processing, we perform some initial relaxation from Gauss-Seidel procedure based on barycentric embeddings.

$$p_i = \sum_{j \in N(i)} w_{ij} p_j \qquad i = 1, 2, \ldots\ldots, n$$

$$w_{ij} = \begin{cases} 1/d_i, & j \in N(i) \\ 0, & j \notin N(i) \end{cases}$$

$$\|p_i\| = 1 \qquad i = 1, 2, \ldots\ldots, n$$

Where, *N(i)* represents the indices for the neighboring vertices of the *i-th* vertex and d_i is the number of elements in *N(i)*.

Here we use uniform mapping method with identical weight for each neighboring vertex. Alexa's (45, 46) parameterization employs such method throughout the whole mapping process until all the foldovers are eliminated. As we tested in some cases, most (over 90%) of the vertices will be displaced without overlapping each other within 100-200 iterations. However, to solve the remaining (~10%) overlapping vertices may cost over 10,000 iterations. Considering the number of vertices involved, these later iteration steps are very expensive and inefficient computationally. And sometimes these iterations can even be endless and the overlapping cannot be removed completely. So, we apply small amount of relaxation iterations here to solve most of the overlapping and employ our following solution to target right on the overlapping area.

Overlapping Solution

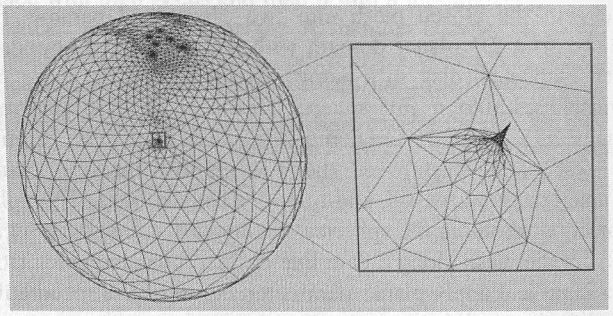


Figure 2: Remaining overlapping spike after initial relaxation

As discussed above, our method emphasizes on parameterization for overlapping areas instead of treating each vertex equally. After the initial relaxation process, most vertices of the mesh are well located except some overlapping spikes for those areas with high populations of vertices, as shown in Figure 2. The relaxation is more or less like a process to drag and expand these areas little by little. With the distribution of highly dense vertices in these areas, the relaxation could be

extremely slow and unreliable. The basic idea of our method is to stretch the boundary of these overlapping areas so that more space will be generated to accelerate the mapping process. The overlapping solution begins with finding overlapping area, followed by stretching such areas. And finally vertices within these areas will be relaxed with fixed boundaries.

Overlapping Identification - This step is to find out the overlapping regions and identify the exact vertices involved. The overlapping vertex is defined as: for a single vertex on the sphere, the line segment between this vertex and the sphere center intersect with any triangle on the sphere. The steps to check overlapping are: i) find out if the line intersects with the plane defined by the three vertices from the triangle; ii) if so, check if the intersection point lies within the line segment with ends of the sphere center and the selected vertex; iii) and if so, check if the intersection point lies within the triangular area.

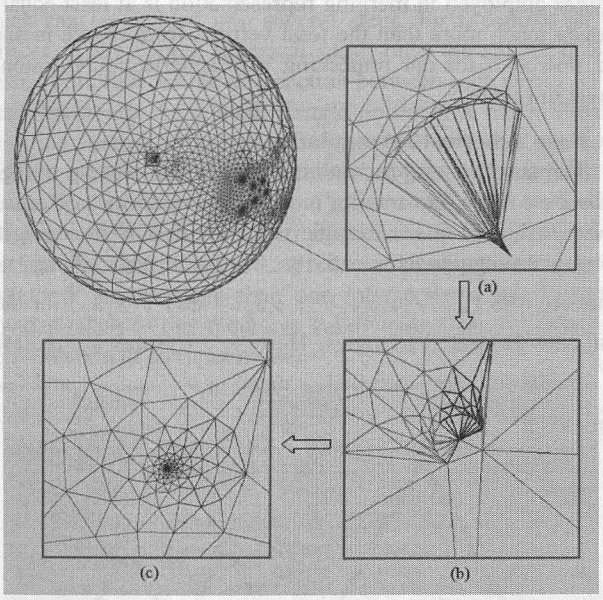


Figure 3: Solution for overlapping: (a) stretched overlapping area; (b) overlapping vertices placed on centroid; (c) overlapping and boundary vertices relaxed

Usually, every single vertex will have to be put into an iteration to make sure if it overlaps with any triangle from the mesh. So for each vertex, we need to keep calculating and find out its relation with every single triangle until an intersection be found, which is quite time-consuming. For example, we test with a spherical triangular mesh with about 2,500 vertices and find out that it takes about 9 seconds to find out all the overlapping vertices. Instead of directly applying this method for overlapping searching, we modify it quite a bit to accelerate the process. We employ a neighborhood searching routine to complete the task. First, loop through all the triangles and check their orientations (spherical surface normal). This will lead to identify the "folded" triangles which have the normal direction toward inside the sphere. Then, based on the vertices from these "folded" triangles, find out their unfolded neighboring vertices and check if they overlap with any of the "folded" triangles. During this process, if none of the neighbors for a selected vertex overlap with any of the "folded" triangles, stop searching around this vertex and keep working with other ones until all the vertices become clear (no overlapping neighbors). Our searching algorithm here is very accurate and performs in a much faster manner. Still with the same mesh with about 2,500 vertices, this searching algorithm can complete the operation with a time scale of milliseconds.

Overlapping Displacement - Overlapping vertices are identified and marked from the previous step. This is followed by displacing overlapping vertices. Before any manipulation applied, the overlapping vertices will be sorted into a collection with nested structure. There usually will be multiple overlapping areas as shown in Figure 2. The sorting will firstly separate these vertices into groups based on their connectivity with each other, which will enable a bunch of connected overlapping vertices into a group. After that, each group of vertices will then be sorted from the shortest distance to outside non-overlapping vertices. This will form several nested groups of overlapping vertices for each spherical triangulation.

The idea of mapping overlapping areas is trying to simplify 3D parameterization problem into several 2D ones. To assist local parameterization for each overlapping area, a stretching process is employed to generate local convex spaces, as shown in Figure 3a. This is a modified Gauss-Seidel procedure which sets the weight for each overlapping vertex to zero as it is involved as a neighbor vertex in calculation. In this process, the non-overlapping vertices will pull and drag the boundary for overlapping area away from the area. A convex or close to a convex shaped boundary will be created after this process. It is followed by dislocation of overlapping boundary, which will ensure the boundary to be convex. As shown in Figure 3b, for each overlapping region, the overlapping vertices are all placed together on the centroid of the convex boundary.

Overlapping Relaxation - For 2D planar parameterization, a regular convex fixed boundary would mostly guarantee an existing bijective mapping, which is the motive of overlapping displacement from previous step. In this overlapping relaxation step, all the non-overlapping vertices and all the vertices on these overlapping boundaries are fixed. Vertices in the overlapping areas are released with Gauss-Seidel iterations until all the overlapping areas disappear, as shown in Figure 3c. The positions for boundary vertices are also recalculated and updated after all the overlappings are eliminated.

Distortion Minimization

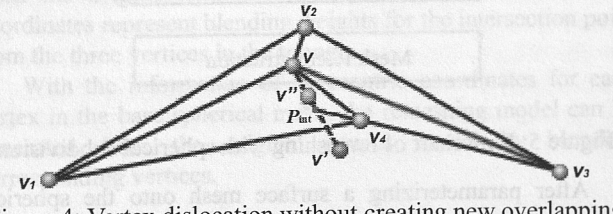


Figure 4: Vertex dislocation without creating new overlapping

Our method focus on solving overlapping with some imposed stretching, which could lead to dislocation for vertices away from their theoretical sites. To minimize distortions created from these dislocations, vertices on the mesh need to be relocated as close to these right positions as possible. While from some testing we have done, directly relax the whole mesh at this moment will bring those overlapping back again.

To minimize these distortions, we implement our minimization method which guarantees bijectivity and avoids overlapping during the process. In our method, we classify all the vertices into two categories: regular and sensitive. The sensitive ones refer to the overlapping vertices we recognized from the previous overlapping identification step. As shown in Figure 4, the current position for vertex P is V and the new position V' is calculated from their neighboring vertices (V_1, V_2, V_3 and V_4) positions. We can easily see that the transition from V to V' takes vertex P outside the region $V_1V_2V_3V_4$, which causes overlapping. Such movement makes the mapping non-bijective and should be prohibited. While, if we skip the movement for vertex P at this step, the movement tendency for this local area may be lost, which can slow down or even prevent local distortion minimization. Here our solution is keeping the same movement with the same path but a smaller scale. As in Figure 4, the scale is defined by VV'', where V'' is the updated new position for vertex P which can be expressed as:

$$V'' = V + (1-\varepsilon)\cdot\overrightarrow{VP_{int}}$$

Here, P_{int} is the overlapping critical position for vertex P which can be calculated from the intersection between VV' and neighbor edge (V_1V_2, V_2V_3, V_3V_4 or V_4V_1).

REMESHING WITH SUBDIVISION

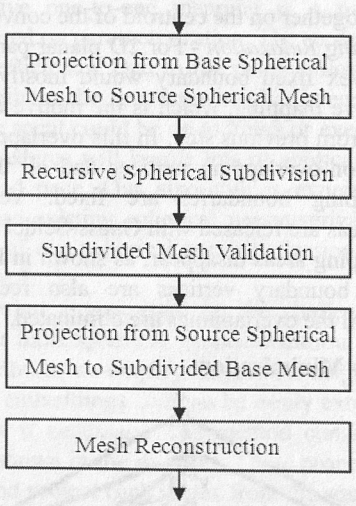


Figure 5: Flowchart of remeshing with spherical subdivision

After parameterizing a surface mesh onto the spherical domain, we now introduce our spherical remeshing method. This method is originally proposed to generate a common connectivity for different mesh models in our mesh morphing framework. We employ the concept of spherical mesh subdivision and extend it in our remeshing algorithm. Spherical mesh subdivision refers to adding details into the spherical triangular mesh by breaking related triangles into smaller ones. Unlike most existing work, our algorithm would subdivide areas which contain detail geometric information rather than subdividing every triangle evenly. This process makes use of few vertices while still cover every geometry detail. Usually, the amount of vertices for remeshing is about the same as the original mesh. And for its application in our mesh morphing framework, the final merged representation has less number of vertices than the sum of all input meshes after aligning the feature areas which typically involve large vertex crowds. Whereas, for most existing morphing methods, the number of vertices employed in merging representation is at least equal to or even much more than the total vertices from source meshes. The flowchart for our remeshing with spherical subdivision is shown in Figure 5.

Base Spherical Triangulations

The subdivision process here is to break existing triangles in the base mesh into smaller ones in the region with details. As shown in Figure 6, the triangle is subdivided by removing the existing triangular connectivity ($v_1v_2v_3$) and adding new connectivities ($v_1v_{12}v_{31}$, $v_2v_{23}v_{12}$, $v_3v_{31}v_{23}$ and $v_{12}v_{23}v_{31}$) for these four new generated triangles. Here v_{12}, v_{23} and v_{31} represent the midpoint for original edges v_1v_2, v_2v_3 and v_3v_1 respectively.

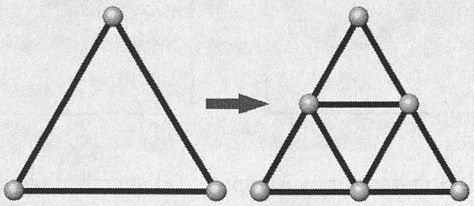
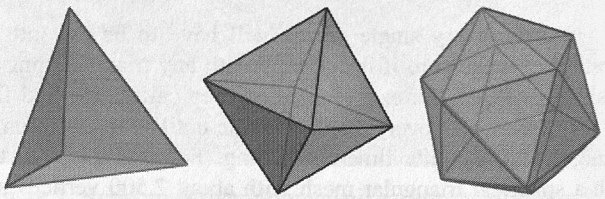


Figure 6: Subdivision by breaking one triangle into four



(a) (b) (c)

Figure 7: Triangular platonic solids: (a) tetrahedron; (b) octahedron; (c) icosahedrons

Since we apply spherical parameterization for closed genus zero mesh, the base subdivision mesh should be equivalent to a sphere topologically. From this point, platonic solids with regular polygon for each face and closed surface are ideal choices. As shown in Figure 7, there are three platonic solids

with triangular faces: tetrahedron, octahedron and icosahedron. We have all three of them available in our algorithm and plan to have further investigation and comparison about the remeshing qualities from different types of these platonic solids in future.

Recursive Spherical Subdivision

To remesh from a source mesh, the algorithm needs to match the level of details (LOD) for the source mesh. This requires the detailed areas (contain dense vertices) to be represented in the remeshing. Here we can utilize a uniform subdivided mesh for this purpose but will involve a large number of vertices. So we implement a local spherical subdivision method which subdivides the triangles in the base mesh to match the corresponding local LOD in the source mesh. In this way, no vertex crowd will be wasted in the open areas which contain little geometric details.

This method employs a recursive subdivision process which can be divided into several steps. First, project all the vertices of the source spherical mesh onto the spherical base mesh. Then, find out the intersection between each vertex of the source mesh and the related triangle on the base mesh. Finally, check the number of hits for each triangle on the base mesh and perform subdivision. If the number of hits for a triangle is greater than a preset value (e.g. 1), the triangle will be broken into four small ones. And each of these four triangles will be checked and subdivided recursively until every triangle in the base mesh has no more than one intersectional hit. Figure 8 provides result of this recursive subdivision from the spherical representation of an input model.

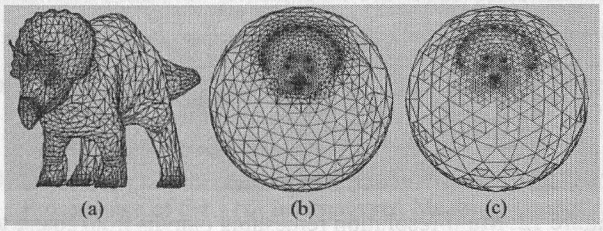


Figure 8: Spherical subdivision: (a) source mesh; (b) spherical mesh; (c) subdivision

Subdivided Triangulation Validation

As we can see, the resulting subdivided base mesh reflects the geometric characteristics of the source mesh. However, the resulting spherical base mesh is not a valid spherical triangulation any more since not all the faces are 3-connected, as shown in Figure 8. To solve this, we implement a triangulation validation process which improves the connectivity without inserting or removing additional vertices.

Invalid triangles occur in the triangles whose neighboring triangles have been subdivided. Middle-vertices for associated edges generated from subdivision cause the problem. For a selected triangle with such issue, the problem can be solved with 3 different situations, as shown in Figure 9. These 3 cases are classified by number of edges that involve in any subdivision. The figure shows how to re-connect these middle-vertices and re-triangulate the corresponding triangle area for different cases.

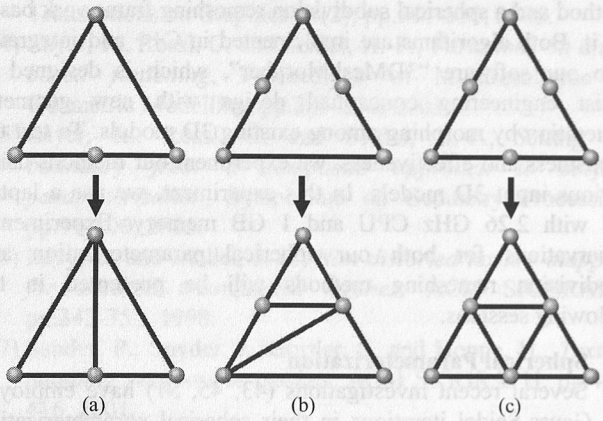


Figure 9: Three cases of validation for a triangle with subdivided neighbor(s)

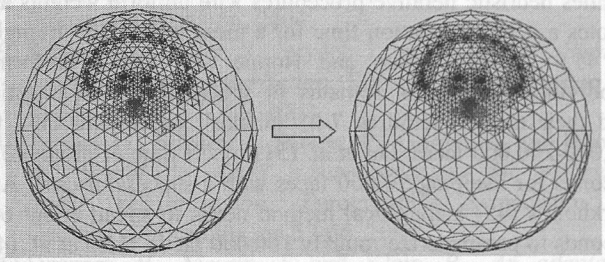


Figure 10: Triangulation validation for a spherical subdivision

Figure 10 presents a result of re-constructed spherical triangulation from an example mesh. We can see the representation matches the details from subdivision and still indicates a valid spherical triangulation.

Mesh Reconstruction

Now, we have a spherical representation for the remeshing after the subdivision and followed validation procedures. From this representation, we want to generate the remeshing model for the original 3D model. To achieve this, we will perform a reverse operation as we does in the subdivision process. Here we project the spherical subdivided mesh back into the spherical mesh for the original model and find out the intersections between vertices from the subdivided mesh and triangles from the original spherical mesh. Then the barycentric coordinates for the intersection point in the associated triangle from the original spherical mesh will be calculated. Such coordinates represent blending weights for the intersection point from the three vertices in the triangle.

With the information of barycentric coordinates for each vertex in the base spherical mesh, the remeshing model can be generated from the original source model by blending corresponding vertices.

EXPERIMENTAL RESULTS

In this work, we present a spherical parameterization method and a spherical subdivision remeshing framework based on it. Both algorithms are implemented in C++ and integrated into our software "3DMeshMorpher", which is designed to assist engineering conceptual design with new geometry generation by morphing among existing 3D models. To test the robustness and effectiveness, we experiment our methods using various input 3D models. In this experiment, we use a laptop PC with 2.26 GHz CPU and 1 GB memory. Experimental observations for both our spherical parameterization and subdivision remeshing methods will be presented in the following sessions.

Spherical Parameterization

Several recent investigations (43, 45, 51) have employed the Gauss-Seidel iterations in their spherical parameterization methods. However, it is reported that their procedures are shown to be unstable and don't guarantee bijectivity. Alexa (45) applies heuristic iterative procedures with uniform weights and quotes a parameterization time for a mesh with 4,169 triangles at 45.9 seconds. Praun and Hoppe (49) utilize uniformly subdivided polyhedral domains in spherical parameterization and their method requires 7-25 minutes' processing time for 25,000-200,000 faces. Gu et al. (51) report time of around 530 seconds for mapping 30,000 faces with a successful case. And Birkholz's (43) hierarchical method needs to spend about 600 seconds to parameterize roughly 100,000 faces. Saba et al. (48) developed a fast numerical solution which could efficiently solve the non-linear equations for spherical mapping from Gotsman et al. (47). They report a minimum total solution time of 8.15 seconds for a model with 5,660 triangles.

Here we test our spherical method with some models and the results are shown in the following Table 1. As we can see, this algorithm can handle thousands of faces in a couple of seconds.

Model	Vertices	Triangles	Weights	**Time (sec)**
horse	1929	3854	uniform	**1.4**
armadillo	2164	4324	uniform	**1.7**
triceratops	2832	5660	uniform	**2.4**
cow	2904	5804	uniform	**2.9**

Table 1: Statistics of spherical parameterization efficiencies

Spherical Remeshing with Subdivision

We have spherical representations for the input meshes from our fast spherical parameterization method. Now, we will test our remeshing algorithm with spherical subdivision. The following Figure 11 shows original source mesh together with the remeshing results from 3 different base meshes: tetrahedron, octahedron and icosahedron. As we can see, they all illustrate the geometric details from the source mesh pretty well with about same amount of vertices.

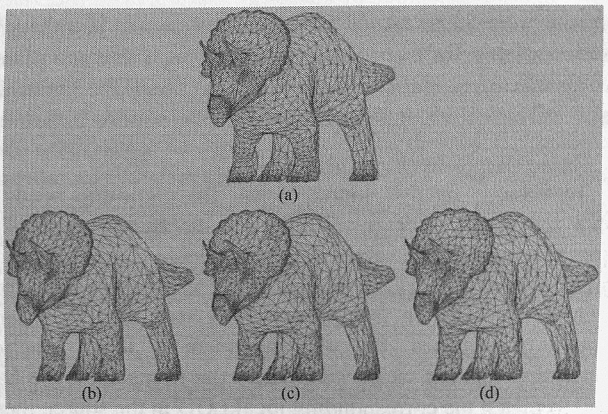


Figure 11: Remeshing from spherical subdivision: (a) source mesh; (b) remeshing with tetrahedron; (c) remeshing with octahedron; (d) remeshing with icosahedrons

As discussed at the recursive spherical subdivision section, we subdivide the base mesh until it matches some defined LOD parameter. By modifying such setting within our algorithm, a bunch of remeshing meshes can be delivered with different resolutions. This capability of multi-resolution can be applied to aid mesh decimation and mesh refinement related applications. Figure 12 shows two examples with LOD from coarse to fine generated with this algorithm.

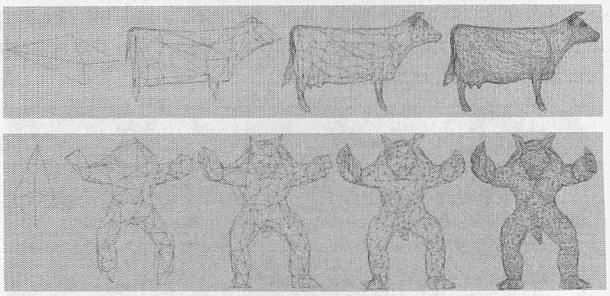


Figure 12: Multi-resolution remeshing outputs from coarse to fine

DISCUSSION AND CONCLUSION

In this paper, we have presented a method to quickly parameterize genus zero polyhedral onto a spherical domain. This algorithm is demonstrated to be faster than existing methods through a collection of 3D models. This framework could enable users to spherically map models with large amount of vertices within a short period of time.

Our parameterization algorithm doesn't theoretically guarantee all overlappings can be solved. While, from our experiments with different models, we don't see such problems. Since our parameterization method targets on solving overlapping, the solution time is more decided by the vertex density of overlapping area instead of the overall number of vertices. The distortion minimization doesn't eliminate all the distortions at once but could reduce the distortion to the desired magnitude if necessary amount of time is allowed. We still have

room to improve our algorithm for better performance. The session of overlapping relaxation will be investigated and improved, which currently is the most time-consuming part in the scheme. We will also investigate different Gauss-Seidel iterations with different weights other than uniform, such as conformal, mean-value and etc.

Based on this spherical parameterization method, we also introduced a robust remeshing framework with spherical mesh subdivision. The method can be applied with a valid spherical representation from input mesh. With about same number of vertices, the remeshing model is able to cover all the geometric details from the original mesh. The LOD of this remeshing process can be adjusted, which leads to a group of mesh representations with different resolutions. This remeshing framework with multi-resolution can be applied to benefit various graphical applications including geometry rendering, mesh simplification, mesh refinement, model morphing and etc.

REFERENCES

(1) Bennis, C., Vezien, J.-M. and Iglesias, G., *Piecewise surface flattening for nondistorted texture mapping*, ACM SIGGRAPH, pp.237-246, 1991.

(2) Floater, M. and Hormann, K., *Surface parameterization: a tutorial and survey*, Advances in Multiresolution for Geometric Modelling, pp.157-186, 2005.

(3) Sheffer, A. Praun, E. and Rose, K., *Mesh parameterization methods and their applications*, Foundations and Trends in Computer Graphics and Vision, 2(2):105-171, 2006.

(4) Eck, M., Derose, T., Duchamp, T., Hoppe, H., Lounsbery, M. and Stuetzle, W., *Multiresolution analysis of arbitrary meshes*, ACM SIGGRAPH, pp. 173-182, 1995.

(5) Kharevych, L., Springborn, B. and Schröder, P., *Discrete conformal mappings via circle patterns*, ACM Transactions on Graphics 25(2), 2006.

(6) Guskov, I., *An anisotropic mesh parameterization scheme*, Proceedings of the 11th International Meshing Roundtable, pp.325-332, 2002.

(7) Floater, M., *Parameterization and smooth approximation of surface triangulations*, Computer Aided Geometric Design, 14(3), pp. 231-250, 1997.

(8) Lee, Y., Kim, H. S. and Lee, S., *Mesh parameterization with a virtual boundary*, Computers and Graphics 26(5), pp.677-686, 2002.

(9) Zhang, E., Mischaikow and Turk, G., *Feature-based surface parameterization and texture mapping*, ACM Transaction on Graphics 24(1), pp.1-27, 2005.

(10) Lévy, B., Petitjean, S., Ray, N. and Maillot, J., *Least squares conformal maps for automatic texture atlas generation*, ACM SIGGRAPH, pp.362-371, 2002.

(11) Desbrun, M., Meyer, M. and Alliez, P., *Intrinsic parameterizations of surface meshes*, Computer Graphics Forum, 21, pp.209-218, 2002.

(12) Sheffer, A. and De Sturler, E., *Parameterization of faceted surfaces for meshing using angle based flattening*, Engineering with Computers 17(3), pp.326-337, 2001.

(13) Sheffer, A., Lévy, B., Mogilnitsky, M. and Bogomyakov, A., *ABF++: fast and robust angle based flattening*, ACM Transactions on Graphics 24(2), pp.311-330, 2005.

(14) Zayer, R., Rossl, C. and Seidel, H.-P., *Variations on angle based flattening*, Proceedings of Multiresolution in Geometric Modelling, pp.285-296, 2003.

(15) Zayer, R., Rössl, C. and Seidel, H.-P., *Setting the boundary free: a composite approach to surface parameterization*, Symposium on Geometry Processing, pp.91-100, 2005.

(16) Lévy, B. and Mallet, J.-L., *Non-distorted texture mapping for sheared triangulated meshes*, ACM SIGGRAPH, pp.343-352, 1998.

(17) Sander, P., Snyder, J., Gortler, S. and Hoppe, H., *Texture mapping progressive meshes*, ACM SIGGRAPH, pp.409-416, 2001.

(18) Zhou, K., Snyder, J., Guo, B. and Shum, H.-Y., *Iso-charts: Stretch-driven mesh parameterization using spectral analysis*, Eurographics Symposium on Geometry Processing, pp.47-56, 2004.

(19) Sander, P., Gortler, S., Snyder, J. and Hoppe, H., *Signal-specialized parameterization*, Eurographics Workshop on Rendering, pp.87-100, 2002.

(20) Tewari, G., Snyder, J., Sander, P., Gortler, S. and Hoppe, H., *Signal-specialized parameterization for piecewise linear reconstruction*, Eurographics Symposium on Geometry Processing, pp.57-66, 2004.

(21) Degener, P., Meseth, J. and Klein, R., *An adaptable surface parameterization method*, Proceedings of 12th International Meshing Roundtable, pp.201-213, 2003.

(22) Maillot, J., Yahia, H. and Verroust, A., *Interactive texture mapping*, ACM SIGGRAPH, pp. 27-34, 1993.

(23) Sander, P., Wood, Z., Gortler, S., Snyder, J. and Hoppe, H., *Multi-chart geometry images*, ACM Symposium on Geometry Processing, 2003.

(24) Gu, X. and Yau, S.-T., *Global conformal surface parameterization*, Symposium on Geometry Processing, pp.127-137, 2003.

(25) Tarini, M., Hormann, K., Cignoni, P. and Montani, C., *PolyCube-Maps*, ACM SIGGRAPH, pp.853-860, 2004.

(26) Sheffer, A. and Hart, J., *Seamster: inconspicuous low-distortion texture seam layout*, IEEE Visualization, pp.291-298, 2002.

(27) Sorkine, O., Cohen-Or, D., Goldenthal, R. and Lischinski, D., *Bounded-distortion piecewise mesh parameterization*, IEEE Visualization, pp.355-362, 2002.

(28) Lazarus, F., Pocchiola, M., Vegter, G. and Verroust, A., *Computing a canonical polygonal schema of an orientable triangulated surface*, In Proceedings of the Seventeenth Annual Symposium on Computational Geometry, 2001.

(29) Erickson, J. and Har-Peled, S., *Optimally cutting a surface into a disk*, Discrete Computational Geometry 31(1), pp.37-59, 2004.

(30) Ni, X., Garland, M. and Hart, J. C., *Fair morse functions for extracting the topological structure of a surface mesh*, ACM SIGGRAPH, pp.613-622, 2004.
(31) Edelsbrunner, H., Letscher, D. and Zomorodian, A., *Topological persistence and simplification*, Discrete and Computational Geometry 28, 4, pp.511-533, 2002.
(32) Guskov, I., Khodakovsky, A., Schröder, P. and Sweldens, W., *Hybrid meshes: multiresolution using regular and irregular refinement*, ACM Symposium on Computational Geometry, pp.264-272, 2002.
(33) Praun, E., Sweldens, W. and Schröder, P., *Consistent mesh parameterizations*, ACM SIGGRAPH, pp.179-184, 2001.
(34) Khodakovsky, A., Litke, N. and Schröder, P., *Globally smooth parameterizations with low distortion*, ACM SIGGRAPH, pp.350-357, 2003.
(35) Schreiner, J., Asirvatham, A., Praun, E. and Hoppe, H., *Inter-Surface Mapping*, ACM SIGGRAPH, 2004.
(36) Kraevoy, V. and Sheffer, A., *Cross-parameterization and compatible remeshing of 3D models*, ACM SIGGRAPH, 2004.
(37) Boier-Martin, I., Rushmeier, H. and Jin, J., *Parameterization of triangle meshes over quadrilateral domains*, Eurographics Symposium on Geometry Processing, pp.197-208, 2004.
(38) Isenburg, M. Gumhold, S. and Gotsman, C., *Connectivity shapes*, Proceedings of IEEE Visualization, pp.135-142, 2001.
(39) Haker, S., Angenent, S., Tannenbaum, S., Kikinis, R., Sapiro, G. and Halle, M., *Conformal surface parameterization for texture mapping*, IEEE TVCG, 6(2), pp.181-189, 2000.
(40) Zayer, R., Rössl, C. and Seidel, H.-P., *Curvilinear spherical parameterization*, Proceedings of Shape Modeling and Applications, pp.57-64, 2006.
(41) Das, G. and Goodrich, M. T., *On the complexity of optimization problems for 3-Dimensional convex polyhedra and decision trees*, Computational Geometry, 8, pp.123-137, 1997.
(42) Shapiro, A. and Tal, A., *Polyhedron realization for shape transformation*, The Visual Computer14 (8), pp.429-444, 1998.
(43) Birkholz. H., *Shape-preserving parametrization of genus 0 surfaces*, Proc. Winter Conference on Computer Graphics (WSCG), 2004.
(44) Kobbelt, L. P., Vorsatz, J., Labisk, U. and Seidel, H.-P., *A shrink-wrapping approach to remeshing polygonal surfaces*, Proceedings of Eurographics, 1999.
(45) Alexa, M., *Merging polyhedral shapes with scattered features*, The Visual Computer, 16(1), pp.26-37, 2000.
(46) Alexa, M., *Recent advances in mesh morphing*, Computer Graphics Forum 21(2), pp.173-196, 2002.
(47) Gotsman, C., Gu, X. and Sheffer, A., *Fundamentals of spherical parameterization for 3D meshes*, ACM SIGGRAPH, pp.358-364, 2003.
(48) Saba, S., Yavneh, I., Gotsman, C. and Sheffer, A., *Practical spherical embedding of manifold triangle meshes*, Proceedings of Shape Modeling International, 2005.
(49) Praun, E. and Hoppe, H., *Spherical parameterization and remeshing*, ACM SIGGRAPH, pp.340-350, 2003.
(50) Sheffer, A., Gotsman, C. and Dyn, N., *Robust spherical parameterization of triangular meshes*, Computing, 72(1–2): 185–193, 2003.
(51) Gu, X., Wang, Y., Chan, T. F., Thompson, P. M. and Yau, S.-T., *Genus zero surface conformal mapping and its application to brain surface mapping*, IEEE Transaction on Medical Imaging 23(7), 2004